DYNAMIC PARAMETER MODIFICATION

Field of the Invention

5   This invention relates to computing systems that are used for the provision of goods, services or information management. It relates particularly to the dynamic modification of parameters existing within such computing systems.

Background of the Invention

10

In modern large computing systems a common topology has three tiers (physical and logical): (i) a presentation tier characterised by multiple workstations focusing on user interactions, (ii) a business tier characterised by multiple servers executing application/business logic, and (iii) a data tier characterised by multiple databases

15   working on data storage and organization. A Local or Wide Area Network (LAN/WAN) interconnects the three tier elements.

Such computing systems find application in many and varied fields, ranging from university research and teaching facilities to business applications. In fact, almost

20   every business will utilise such a system to transact its functions and serve its clients. For example, a system may be used to control inventory, for word processing and accounts purposes, and for servicing client's enquiries. Many businesses have very large client bases and may provide an extensive inventory of goods and services. One illustrative example is a telecommunications service provider (Telco) that serves a countrywide client

25   base. The Telco's subscribers thus can number in the millions, and each customer will expect an immediate response from a Customer Service Representative (CSR) to any inquiry, which can range from billing information, a request for a new service, or the placing of orders for a product.

30   Similar examples are seen in Utilities, insurance companies, banks, hospitals, law firms, accountancy firms, stock exchanges, universities and Government agencies, to name but a few.

The data building blocks of the computing system - at a meta level - are termed

35   'parameters'. These parameters are mapped to instances of customised objects upon

which the system executes the various applications that are supported. Telcos, for example, can have a very large number of parameters that are either externally configured by the customer or by a CSR. Such parameters can include tax rates, customer type, bank codes, and any other information specific to the individual customer's business. There is a need for new parameters to be added, and the structure of existing parameters to be modified, as quickly as possible.

In the prior art, system parameters are typically maintained via a discrete computer program. Every time a new parameter is introduced or the structure of an existing parameter changed, the computer program has to be rewritten in part to accommodate the modifications to the parameter set. Such rewriting requires the skills of a computer programmer and incurring cost and lost time.

It is therefore an object of the present invention to provide for ease of creation and modification of system parameters.

Summary of the Invention

The invention provides a method for maintaining parameter data in a computing system, comprising the steps of:

  (a)   creating a text file of parameter meta data;

  (b)   representing said text file as a Graphical User Interface (GUI) having parameter fields, and for each parameter field, having one or more attribute sub-fields, each of said sub-fields being text editable; and

  (c)   storing attribute text entered in any said sub-field to a data store.

The invention further provides a method for maintaining parameter data in a computing system, comprising the steps of:

  (a)   creating a text file of parameter meta data using XML;

  (b)   representing said text file as a Graphical User Interface (GUI) having parameter fields, and for each parameter field, having one or more attribute sub-fields, each of said sub-fields being text editable by use of Java code;

  (c)   for a subset of said text file corresponding to a parameter, calling any existing data from said data store for said the relevant sub-fields; and

Examples of parameters included tax rates, customer types, bank codes, and any other information specific to a user or customer.

Brief Description of the Drawings

Non-limiting embodiments of the invention will now be described with reference to the accompanying drawings, in which:

Fig. 1 is a representative topology of a three tier computing system upon which the invention may be practised;

Fig. 2 shows a block diagram relating to the parameter maintenance process;

Fig. 3 shows a block diagram similar to Fig. 2;

Fig. 4 shows a flow diagram illustrating the parameter maintenance process; and

Fig. 5 shows a GUI representation of the parameter maintenance process.

Detailed Description of Embodiments and Best Mode

Fig. 1 is a representative topology of a three tier computing system 10 embodying the invention. The presentation (or client/user) tier is represented by a number (1...n) of workstations 20, that can be appropriate computing terminals, for example personal computers. The business tier is represented by a number (1...p) of servers 30 , that can be dedicated mini or mainframe computers. The data tier is represented by a number (1...m) of databases 40, which can include dynamically managed magnetic or optical storage media.

The computing system 10 is of an 'open' design, providing communication links 60,62,64, via external networks 70,72,74 to like-devices 22,32,42 and remote telephone terminals 24, 26.

The workstations 20, servers 30, and databases 40 are interconnected by a Local or Wide Area Network (LAN or WAN) 50. The LAN/WAN 50 carries information passing between each of the three basic elements described. It will be appreciated that the topology shown in Fig. 1 is representative only, and that any other convenient form of

network could be implemented such that the objective of information passing between the workstations 20, servers 30 and databases 40 is achieved.

Embodiments of the invention find industrial application in the fields noted in the foregoing Background section. For the purposes of a non-limiting illustration, consider the example of a Telco operating across many States of the United States. Such a Telco will typically support local, regional, interstate and international voice and data calls, as well as cellular mobile voice and data traffic. Customers of the Telco can choose from a wide range of goods and services including, for example, the installation of second phone/fax/Internet lines, call forwarding, and messaging. They also will expect to be able to make enquiries of CSRs stationed at the workstations 20 concerning billing and service faults. It is not unreasonable to expect a modern-day Telco to have at least 1 million customers, typically requiring at least 500 CSRs. A Telco system infrastructure of this size can expect to handle about 15,000 business transactions per hour. For each business transaction there may be 6 CSR machine transactions required, and each individual machine transaction can involve up to 20 database transactions (i.e. I/Os).

To give a better example of the size of computing hardware required to achieve such performance, it is considered that the CSR workstations 20 should be Pentium™ personal computers running the Windows NT™ operating system, the servers 30 can be one or more IBM UNIX™-based 12-way RS6000™ S-70 machines, and the databases would require a capacity of about 40 Gbytes, managed by an Oracle ™ or IBM DB-2™ system. There would, of course, be other operational LAN/WAN servers required to handle data communications, as would be readily understood by a person skilled in the art.

In business systems such as Telcos, customers call CSRs and request goods or services in everyday language, such as a request for 'call waiting' to be activated on a domestic telephone line. Indeed, the CSR also operates at this level and is presented with information (as a GUI) on the display of the workstation relating to such goods and services. The computing system 10 then acts on customers ordered goods or services or enquiries.

It is usual also to introduce the ideas of 'Definition Time', 'Maintenance Time' and 'Order Management Time'. Definition Time relates to the definition of a new

product to the point where it is available for association with a customer. Maintenance Time is where an existing Product is modified in terms of its Constituent Actions, Features and Tasks. Order Management Time is when the set of Products are alive and available to the CSRs at the workstations.

Before a specific embodiment is described, it is important to appreciate that the invention has broad application to computing systems that utilise instances of objects as data blocks, that are only statiated at run-time before being passed to between workstation, server or database.

The data exists at all other times at a meta-level as generic parameters having sets or fields of attributes. For the present invention, the parameters exist as Extensible Markup Language (XML) text.

Telco Example

The 'goods and services' supported by the system 10 are termed "Products". Products exist within both the workstations 20 and the servers 30 of the computing system 10. Products are defined during Definition Time to be available for a CSR to populate an instance in response to a customer order. The Products are essentially a 'business world' construct.

Products can be defined to suit a particular business or customer need. In that sense, an existing Product can be amended in some way, or a new Product can be added to a 'Product Book', which is a logical grouping of Products, such as those that can be sold to residential customers. In the Telco environment, Products might include:

Standard Wireline Access: The wireline connection into a home, allowing the home to have the physical product of a telephone. It is the line into the house termining in the adaptor that sits on the wall that telephones plug into.
Call Waiting: The ability to notify a customer that they have another call coming in to their phone line when they are on the phone.
Call Diversion: The ability to re-route telephone calls to another phone/mail box/pager, etc.

In the 'system world', it is the 'parameters' that represent the data building blocks that are acted upon by the computing application. Parameters are constituted by 'attributes'.

Parameters can be modified in any one of the Definition Time, Maintenance Time and Order Management Time.

Parameter Maintenance

Fig. 2 shows a block diagram of the parameter maintenance application. The application Graphical User Interface (GUI) 150 (residing on a workstation 20) has two principal sub-fields: a navigator panel 152 and an edit panel 154. An XML textual file 160 is written to the application GUI 150 by Java™ code. The Java code resides on a server 30 in the sense of Fig. 1. Additionally, data is retrieved or stored from the database 170 to/from the GUI 150 in a manner that will presently be described. In the sense of Fig. 1, the database 170 would be one of the databases 40 on the WAN/LAN 50.

Fig. 3 is a block diagram similar to Fig. 2. The XML file 160 contains meta data of parameters arranged as groups. The navigator panel 152 shows, for the XML file, a corresponding tree structure having groups of parameters. Within each group is a set of parameters 184, noted as P1 and P2. When a particular parameter (P1) is selected for maintenance, it was then caused to appear in the edit panel 154 where its (textual) attributes can be populated or modified. The rows in the edit panel 54 thus represent individual parameters, while the columns represent the attributes of each parameter.

The maintenance process will now be described further with reference to Fig. 4, which is a flow diagram.

In step 200 the maintenance application commences. The XML file with the parameter meta data of step 202 is read in step 204. The XML meta data is parsed into an internal tree representation in step 206. The tree representation is constructed and displayed as the GUI panels 152, 154 in step 208. The user then selects a node on the GUI tree that appears in the navigator panel 152 in step 210. In step 212, a retrieve operation is performed from the parameter database 214. The retrieved parameter records are displayed in step 216 in tabular form in the edit panel 154. Now the user is able to

modify the record data, in step 218 for the displayed parameter (P1) which is then stored to the parameter database 214. In step 220 it is determined whether the maintenance procedure is finished and, if so, the maintenance application terminates in step 220, else a further node is selected in the navigator panel 152 in step 210.

5

An example of an XML file, omitting some repetitive 'attribute name' detail, is as follows:

```
<?xml version="1.0?" >
<!DOCTYPE collection SYSTEM "parameters.dtd">
<collection name="Collection" >
 <common name="Common" >
  <datastore name="Store"uri="serialized-parameter-file:///C:\temp\">
  <attribute-set name="Prefixed" >
   <attribute name="Activation" id="Activation"type="Data"/>
   <attribute name="SortOrder" type = "String"/ >
   </attribute-set>
   <attribute-set name="Postfixed">
    <attribute name="Deactivation" id="Deactivation" type="Data"/>
  </attribute-set >
  <attribute-set name=Audit>
  </attribute-set>
 <common>
 <group name="Billing">
 <group>
 <group name="Customer">
   <parameter id="TCUSTYPE"=name="Customer Type" store-name="TCUSTYPE" >
   <attribute is Key="yes" name="Type" type = "Sring" id="TYPE store-name" CTYPE" >
<attributename="Description"type="String"id="DESCRIPTION"store-name=(DESCRIPTION"/>
   <attribute name="Discount" type="Integer"id="DISCOUNT' store-name=(DISCOUNT"/ >
   <parameter>
   <parameter id="TACCDEF" name="Account Default" store-name="TACCDEF" >
    <attribute name="Customer type" type="String" id="CUSTTYPE"
        store-name="CUSTTYPE"/>
    <attribute name="Customer subtype" type="String"id="CUSTSUBTYPE"
        store-name="CUSTTYPE:/>
    <attribute name="Detailed bill option" type="String" id="DETAILEDBILLOPTION"
        store-name="CUSTSUBTYPE"/ >
    <attribute name="Current charges summary"type="String" id="CURRENT CHARGESSUMM"
    store-name="CURRENT CHARGESSUMM"/>
    <attribute name="Print S&E by line" type="String" id="PRINTSUMMBYLINE"
        store-name="PRINTSUMMBYLINE"/ >
    <attribute name="Refund required" type="String" id="REFUNDREQUIRED"
        store-name="REFUNDREQUIRED"/ >

                                     *
                                     *
                                     *

    <attribute name="UpdateIdentifier" type="String" id="UPDATEID"
        store-name="UPDATEID"/>
   </parameter>
  </parameter id="TADDRPART" name="Address Part" store-name="TADDRPART">
   <attribute name="Address part type code" type="String" id="ADDRPARTTYPECODE"
        store-name="ADDRPARTTYPECODE"/>
```

```
<attribute name="Address part type Description" type="String" id="ADDRPARTTYPEDESC" >
    store-name=" ADDRPARTTYPEDESC"/>
        <attribute name="Update ID" type="String" id="UPDATEID"
            store-name="UPDATEID"/>
        </parameter>
```

Fig. 5 is a more detailed GUI representation showing the navigator panel 152 and the edit panel 154 for the XML file noted above. The edit panel is made up of two sub-fields: the upper panel 156 representing a line summary of the attributes pertaining to each parameter, and the lower field 158 providing for imput fields for a user. In the example shown, the parameter is Account default and a portion of the parameters are shown in the navigator panel 152. Specifically, the attributes are: customer type, customer sub-type, detailed bill option, current charges summary, and refund required. All of these attributes can be noted in the preceding listing.

Parameter data is stored in a table, which may be a table within a traditional database or may be in a binary file stored in the file system (i.e. databases 40).

The structure of the parameter is represented in XML. Each parameter is identified by elements surrounded by "parameter" tags. A "tag" is a word that has meaning as a delimiter in XML. Within the attribute definition, the parameter key (or code) attribute is identified by the signature "isKey=yes" appearing in the attribute definition.

The storage table may be a table stored in a database or may be in a binary file stored in the file system. Either way, a parameter has a number of parameter instances each represented a row in a table data structure and each uniquely identified by a key attribute. The columns in the table correspond to individual parameter attributes.

An example would be a "Customer Type" parameter which may have an attribute named "Type" representing the parameter code and a number of other attributes including a "Description" attribute. Each record in the parameter storage table will have a different value for the code attribute. The following is a schematic representation of the storage table for a simple "Customer Type" parameter:

Type	Description	Discount
"RES"	"Residential"	5

"BUS"   "Business"   10

Any attribute of a parameter can be modified as a screen-based edit function. Such a modification is then stored in a data store by the use of a URI included in the XML file. The URI is structured so as to specify the database protocol before the first colon, and thereafter protocol specific location parameters are specified. In the present example, the application can determine that the datastore is a serialized file located within the file system within directory "C:\temp". An individual parameter table can be located by appending the parameter identifier to the datastore URI.

The intention can be utilised to edit/amend an existing attribute or the populate vacant attribute sub-fields. In a similar way, the invention can add or modify parameters by simple amendment of the relevant XML file.

Another Embodiment

The invention also can be utilized on a stand-alone computing device, such as a Personal Computer (PC). The PC may be running a database application program within which there are multiple parameters that are required to be modifyable. The invention functions in an identical manner to that described above, except that the database storage device (e.g. the database 170 shown in Fig. 2) resides locally with the stand-alone PC.

Advantages of the Invention

The invention has advantages over the prior art by eliminating the need to create or modify application code with the introduction or modification of parameter types. Instead, an XML file can be edited and reloaded while the relevant system application is still running, and the maintenance GUI dynamically re-generated to accommodate modifications and allow new parameter values to be entered.

Dynamic calls are made by the maintenance application to create the required database tables or binary files, instead of requiring a skilled resource (e.g. computer programer or system operator) to do this manually.

The UI is dynamically generated and new values can be entered against the new or modified parameters without requiring a restart of the maintenance application and without the need for a skilled developer to build and test the UI.

5   It is to be understood that the invention is not limited only to the embodiments described, but that numerous alterations and modifications, will be apparent to one skilled in the art, are included within its scope.